LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-TR-423702

Petascale Computing Enabling Technologies Project Final Report

B. R. de Supinski

February 16, 2010

Disclaimer

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Petascale Computing Enabling Technologies Project Final Report

The Petascale Computing Enabling Technologies (PCET) project addressed challenges arising from current trends in computer architecture that will lead to large-scale systems with many more nodes, each of which uses multicore chips. These factors will soon lead to systems that have over one million processors. Also, the use of multicore chips will lead to less memory and less memory bandwidth per core. We need fundamentally new algorithmic approaches to cope with these memory constraints and the huge number of processors. Further, correct, efficient code development is difficult even with the number of processors in current systems; more processors will only make it harder.

The goal of PCET was to overcome these challenges by developing the computer science and mathematical underpinnings needed to realize the full potential of our future large-scale systems. Our research results will significantly increase the scientific output obtained from LLNL large-scale computing resources by improving application scientist productivity and system utilization. Our successes include scalable mathematical algorithms that adapt to these emerging architecture trends and through code correctness and performance methodologies that automate critical aspects of application development as well as the foundations for application-level fault tolerance techniques.

PCET's scope encompassed several research thrusts in computer science and mathematics: **code correctness** and **performance methodologies**, **scalable mathematics algorithms** appropriate for multicore systems, and application-level **fault tolerance** techniques. Due to funding limitations, we focused primarily on the first three thrusts although our work also lays the foundation for the needed advances in **fault tolerance**.

In the area of **scalable mathematics algorithms**, our preliminary work established that OpenMP performance of the AMG linear solver benchmark and important individual kernels on Atlas did not match the predictions of our simple initial model. Our investigations demonstrated that a poor default memory allocation mechanism degraded performance. We developed a prototype NUMA library to provide generic mechanisms to overcome these issues, resulting in significantly improved OpenMP performance. After additional testing, we will make this library available to all users, providing a simple means to improve threading on LLNL's production Linux platforms. We also made progress on developing new scalable algorithms that target multicore nodes. We designed and implemented a new AMG interpolation operator with improved convergence properties for very low complexity coarsening schemes. This implementation will also soon be available to LLNL's application teams as part of the *hypre* library. We presented results for both topics in an invited plenary talk entitled ``Efficient Sparse Linear Solvers for Multi-Core Architectures" at the 2009 HPCMP Institutes Annual Meeting/CREATE Annual All-Hands Meeting. The interpolation work was summarized in a talk entitled ``Improving Interpolation for Aggressive Coarsening" at the 14th Copper Mountain Conference on Multigrid Methods and in a research paper that will appear in Numerical Linear Algebra with Applications [1].

In the area of **code correctness**, we significantly extended our behavior equivalence class identification mechanism. Specifically, we not only demonstrated it works well at very large scales [2] but we also added the ability to classify MPI tasks not only by function call traces, but also by specific *call sites* (source code line numbers) being executed by tasks. More importantly, we developed a new technique to determine relative logical execution progress of tasks in the equivalence classes by combining static analysis with our original dynamic approach. We applied this technique to a correctness issue that arose at 4096 tasks during the development of the new AMG interpolation operator discussed above. This scale isat the limit of effectiveness of production tools, but our technique quickly located the erroneous source code, demonstrating the power of understanding relationships between behavioral equivalence classes. This work is the subject of a paper recently accepted to SC09 [3], as well as a presentation entitled ``Providing Order to Extreme Scale Debugging Chaos'' given at the ParaDyn Week annual conference in College Park, MD. In addition to this theoretical extension, we have made significant progress in developing a front end for this tool set, and the front-end is now available on several of LLNL's large-scale computing resources. In addition, we explored mechanisms to identify exact locations of erroneous MPI usage in application source code. In this work, we developed a new model that led to a highly efficient algorithm for detecting deadlock during dynamic software testing. This work was the subject of a well-received paper at ICS 2009 [4]. Finally, we have worked with the University of Wisconsin to develop key infrastructure to apply Cooperative Bug Isolation (CBI) to MPI applications. Specifically, we have developed a ROSE-based instrumentor that will allow us to apply this technique to key LLNL applications such as KULL. Preliminary tests with this infrastructure indicate that it has relatively low overhead. As this work extends CBI instrumentation to C++ for the first time, several unique issues arose related to objected-oriented code. These are the subject of a paper currently in preparation.

In the area of **performance analysis**, we developed and extended a wavelet-based technique to gather load balance data scalably [5]. We demonstrated that the mechanism works at 16K tasks on BG/P for real applications. We also applied the technique to PF3D and began to identify code sections that contribute to imbalances in large-scale runs on our production Linux platforms. As part of this work, we developed a front-end for the tool that lets developers scalably visualize load balance of specific code regions on LC machines. The front-end exploits the wavelet representation to display 3D models of massively parallel data efficiently, and it can perform cluster analysis to discover regions of code with similar behavior. The GUI is built on top of VTK, which will enable us to exploit sophisticated visualization techniques such as volume rendering to correlate performance data directly with application data and to optimize model partitioning. This work was the subject of Todd Gamblin's dissertation [6], which he successfully defended while working as an intern on PCET. His work has also had impact on other PCET-related projects, including the creation of a prototype library for scalable call path tracking in the presence of dynamic loading that is currently being used in our scalable MPI fault injection work.

In the area of **fault tolerance**, our initial work in developing models of how soft faults propagate through application code led to Greg Bronevetsky’s successful Early Career Investigator Award from the Office of Science. These models use fault injection to

characterize the impact of soft faults on short regions of application code. These characterizations take the form of erro patterns that can be emulated for repeated testing of the impact of faults on larger regions of the code. We plan to combine this mechanism with compiler-based analyses that capture the flow of faults through application code to determine code regions that are particularly vulnerable to errors. We will then develop techniques to protect those vulnerable regions.

Overall, PCET successfully met many of the challenges arising in future large-scale systems. Perhaps more importantly, it has laid the groundwork for further advances that will ultimately substantially increase the overall usability of those systems.

References:

[1] U.M. Yang. On Long Range Interpolation Operators for Aggressive Coarsening. To appear in *Numerical Linear Algebra with Applications*. Available as LLNL Technical Report LLNL-JRNL-413051, May 2009.

[2] G.L. Lee, D.H. Ahn, D.C. Arnold, B.R. de Supinski, M. Legendre, B.P. Miller, M. Schulz, and B. Liblit. Lessons Learned at 208K: Towards Debugging Millions of Cores. In *SuperComputing 2008 (SC08)*, Austin, TX, Nov. 2008.

[3] D.H. Ahn, B.R. de Supinski, I. Laguna, G.L. Lee, B. Liblit, B.P. Miller, and M. Schulz. Scalable Temporal Order Analysis for Large Scale Debugging. In *SuperComputing 2009 (SC09)*, Portland, OR, Nov. 2009.

[4] T. Hilbrich, B. de Supinski, M. Schulz, and M. Mueller. A Graph Based Approach for MPI Deadlock Detection. In *International Conference on Supercomputing (ICS 2009)*, June 2009.

[5] T. Gamblin, B.R. de Supinski, M. Schulz, R.J. Fowler, and D.A. Reed. Scalable Load-Balance Measurement for SPMD Codes. In *Supercomputing 2008 (SC'08)*, Austin, TX, Nov. 2008.

[6] T. Gamblin. Scalable Performance Measurement and Analysis. PhD thesis, University of North Carolina at Chapel Hill, Chapel Hill, NC, Aug. 2009.